Term-Work 6

Problem Definition:

Simulate a full duplex connection in a wired network using ns3

Objectives:

→ To design and simulate the network in latest simulation.
→ To perform the real time network traffic analysis using network monitoring tools

Theory:

A first ns3 script: If you downloaded the system as suggested, you will have a release of ns3 in a directory called repos under your home directory. Change into release directory. Change into the examples / tutorial directory. You should see a file named first.cc located here. This is a script that will create a simple point to point link between 2 nodes and also a single packet

Boilerplate: The first line in the file is an emacs mode line. This tells emacs about the formatting convention we use in our source code. The ns3 simulator is licensed using the GNU general public license

Module Includes: The code proper starts with a number of include statements

```
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
```

./waf -d debug configure is used to configure the project to perform debug builds.

./waf to build the project

Ns3 namespace : The nextline in first.cc script is a namespace declaration using namespace ns3

Logging : The next line of script is the following
NS_LOG_COMPONENT_DEFINE ("firstscriptexample");

Topology Helpers :

→ Nodecontainer : The next 2 lines of code in our script will actually create the ns3 node objects that will represent the computers in the simulation.
NodeContainer nodes; nodes.create(2);

→ Pointtopointhelper: The next lines in script are, the first line, instianticates a pointpointhelper object to use the stack. The next line, tells the pointtopoint-helper object to use the value "5 Mbps" as the datarate. The next line, tells the pointtopointhelper object to use the value "2 ms" as the value of the transmission delay.

→ Netdevicecontainer : NetDeviceContainer (devices);
devices = pointTopoint.Install(nodes);
These nodes will finish configuring the devices and channel.

→ Internetstackhelper : It is a topology helper that is to internet stacks what the pointtopointhelper net devices.

→ IPv4 addresshelper : The next 2 lines of code in first.cc, declare an address helper object and tell it that it should begin allocating IP addresses from the network 10.1.1.0 using the mask 255.255.255.0 to define the allocatable bits. The next line performs actual address assignment.

→ udpechoserverhelper : These lines are used to set up a UDP echo server application on one of the nodes previously created

→ udpechoclienthelper : The echo client application is set up in a method substantially similar to that for the server. There is an underlying udpechoclientapplication that is managed by an UdpEchoclienthelper

→ Simulator : This is done using the global function
Simulator :: Run ();

Source code:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE("firstscriptexample");
int main(int argc, char *argv[])
{
    CommandLine cmd;
    cmd.Parse(argc, argv);
    Time::SetResolution(Time::Ns);
    LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
    NodeContainer nodes;
    nodes.Create(2);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
    NetDeviceContainer devices;
    devices = pointToPoint.Install(nodes);
    InternetStackHelper stack;
    stack.Install(nodes);

    IPv4AddressHelper address;
    address.SetBase("10.1.1.0", "255.255.255.0");
    IPv4InterfaceContainer interfaces = address.Assign(devices);
```

```
UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));
UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 9);
echoClient.SetAttribute("MaxPackets", UintegerValue(1));
echoClient.SetAttribute("Interval", TimeValue(seconds(1.0)));
echoClient.SetAttribute("PacketSize", UintegerValue(1024));
ApplicationContainer clientApps = echoClient.Install(nodes.Get(0));
clientApps.Start(seconds(2.0));
clientApps.Stop(seconds(10.0));
AnimationInterface anim("first.xml");
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("first.tr"));
pointToPoint.EnablePcapAll("first");
Simulator::Run();
Simulator::Destroy();
return 0;
```

3.

Conclusion:

We learnt to simulate a full duplex connection in a wired network using ns3

References:

→ Https://ns3simulator.com.

→ Https://www.nsnam.org.

Term-Work 7

Problem Statement:

Simulate a full duplex connection in a wired network using ns3.

Theory:

A ns3 script

If you download the system as suggested, you will have a release of ns3 in a directory called repos under your home directory. Change into "release" directory. Change into the examples / tutorial directory. You should see a file named "second.cc" located there. This is a script that will create a simple point-to-point link between 2 nodes.

Boiler plate:

The first time in the file is an emacs mode line. This tells emacs about the formatting we use in our source-code. The ns3 simulator is licensed using the GNU general public license.

Module Includes

The code properly starts with a number of include statements

```
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/netanim-module.h"
#include "ns3/helper-module.h"
```

./waf-d debug configure is used to configure project to perform "debug" builds.
./waf to build the project.

Source code:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
int main (int argc, char *argv[])
{   bool verbose = true;
    uint32_t nCsma = 3;
    CommandLine cmd;
    cmd.addValue ("nCSMA", "Number of \"extra\" CSMA
nodes/devices", nCsma);
    cmd.AddValue ("verbose", "Tell echo applications to log
if true", verbose);
    cmd.Parse (argc, argv);
    if (verbose)                                        INFO),
    {   LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL-
    }                                                   INFO
    nCsma = nCsma == 0 ? 1 : nCsma;
    NodeContainer p2pNodes;
    p2pNodes.Create (2);
    NodeContainer csmaNodes;
    csmaNodes.Add (p2pNodes.Get (1));
```

```
csmaNodes.create (nCsma);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("Datarate", StringValue ("5MLps"));
pointToPoint.SetDeviceAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);
CsmaHelper csma;
csma.setChanelAttribute ("DataRat", StringValue ("100MLps"));
csma.setChanelAttribute ("Delay", TimeValue (Nanosec (6560)));
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);
InternetStackHelper stack;
stack.Install (p2pNodes.Get(0));
stack.Install (csmaNodes);
IpV4AddressHelper address;
address.setBase ("10.1.1.0", "255.255.255.0");
IPV4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
address.setBase ("10.1.2.0", "255.255.255.0");
IPV4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.
Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress
(nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
```

Term-Work 8

Problem Statement:

Simulate a simple 5G network application using NS3

Theory:

A ns-3 script:

If you download the system as suggested, you will have a release of ns-3 in a directory called repos under your home-directory. Change into release directory and then into example/ tutorials directory. You should see a file third.cc located there. This is the script that will simulate a simple 5G network

Boiler plate:

The first line is an emacs mode line. This tells emacs about the formatting conventions we use in our source-code. The ns-3 simulator is licensed using the GNU general public license.

Module Includes

The code properly states a number of include statements

```
#include "ns3/core-modules"
#include "point-to-point-module.h"
#include "network-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
```

./waf -d debug configure is used to configure the project to perform debug builds.

./waf to build project.

Source code:

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
int main (int argc, char *argv[])
{
    bool verbose = true; tracing = false;
    uint32_t nCsma = 3, nWifi = 3;
    CommandLine cmd;
    cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/
devices", nCsma);
    cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
    cmd.AddValue ("verbose", "Tell echo applications to log if
true", verbose);
    cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
    cmd.Parse (argc, argv);
    if (nWifi > 18)
    {   std::cout << "nWifi should be 18 or less; otherwise g
layout exceeds the bounding box" << std::endl;
        return 1;
    }
    if (verbose)
```

```
{ LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
NodeContainer p2pNodes;
p2pNodes.Create (2);
PointToPointHelper pointTopoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);
NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);
NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());
WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac", "Ssid", SsidValue (ssid),
"ActiveProbing", BooleanValue (false));
```

```
NetDeviceContainer staDevices;
staDevices = wifi.Install(phy, mac, wifiStaNodes);
mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid));
NetDeviceContainer apDevices;
apDevices = wifi.Install(phy, mac, wifiApNode);
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator", "MinX",
DoubleValue(0.0), "MinY", DoubleValue(0.0), "DeltaX", DoubleValue(5.0
"DeltaY", DoubleValue(10.0), "GridWidth", UintegerValue(3), "LayoutType"
StringValue("RowFirst"));
    mobility.SetMobilityModel("ns3::MobilityModel", "Bounds",
RectangleValue(-50, -50, -50, -50)));
    mobility.Install(wifiStaNodes);
    mobility.SetMobilityModel("ns3::ConstantPosition");
    mobility.Install(wifiApNode);
    InternetStackHelper stack;
    stack.Install(csmaNodes);
    stack.Install(wifiApNode);
    stack.Install(wifiStaNodes);
    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer p2pInterfaces;
    p2pInterfaces = address.Assign(p2pDevices);
    address.SetBase("10.1.2.0", "255.255.255.0");
    Ipv4InterfaceContainer csmaInterfaces;
    csmaInterfaces = address.Assign(csmaDevices);
    address.SetBase("10.1.3.0", "255.255.255.0");
    address.Assign(staDevices);
    address.Assign(apDevices);
    UdpEchoServerHelper echoServer(9);
```

```
ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));
UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(nCsma), 9);
echoClient.SetAttribute("MaxPackets", UintegerValue(1));
echoClient.SetAttribute("Interval", TimeValue(1));
echoClient.SetAttribute("PacketSize", UintegerValue(1024));
ApplicationContainer clientApps = echoClient.Install(wifiStaNodes.Get
(nWifi-1));
clientApps.Start(Seconds(2.0));
clientApps.Stop(Seconds(10.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
Simulator::Stop(Seconds(10.0));
if(tracing == true)
{
    pointToPoint.EnablePcapAll("third");
    phy.EnablePcap("third", apDevices.Get(0));
    csma.EnablePcap("third", csmaDevices.Get(0), true);
}
AnimationInterface anim("third.xml");
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("third.tr"));
pointToPoint.EnablePcapAll("third");
Simulator::Run();
Simulator::Destroy(); return 0;
}
```

Conclusion: We learnt to simulate a full duplex connection in a wired network using ns3

References:

→ https://www.nsnam.org/ns-3-simulator.com

Term-Work 9

Problem Statement:

Understanding the working of IPV6 in low power lossy network

Theory:

Low power and lossy networks (LLN) are resource contrained. Routers are usually limited in terms of processing power, battery and memory and their interconnects are characterized by unstable links with high loss rates, low data rates and low packet delivery rates.

The traffic patterns could be P2P or P2MP or MP2P. Lossy means the packet drop rate will be high.

Forwarding and Routing:

Up routes towards nodes of decreasing rank (parents), down routes towards nodes of increasing rank. Nodes inform parents of their presence and reachability to descendants.

Source Code:

```
#include "contiki.h"
#include "net/routing/routing.h"
#include "random.h"
#include "net/netstack.h"
#include "net/ipv6/simple-udp.h" #include "net/ipv6/uip-ds6.h"
#include "net/ipv6/uipbuf.h" #include "sys/log.h"
#define LOG_MODULE "App"
#define LOG_LEVEL LOG_LEVEL_INFO
PROCESS(ipv6_hooks_process, "IPV6 Hooks");
AUTOSTART_PROCESSES(&ipv6_hooks_process);
static enum netstack_ip_action
ip_input(void)
{   uint8_t proto = 0;
    uipbuf_get_last_header(uip_buf, uip_len, &proto);
    LOG_INFO("Incoming packet proto: %d from", proto);
    LOG_INFO_6ADDR(&UIP_IP_BUF->srcipaddr);
    LOG_INFO_("\n");
    return NETSTACK_IP_PROCESS;
}
static enum netstack_ip_action
ip_output(const linkaddr_t *localdest)
{   uint8_t proto;
    uint8_t is_me = 0;
    uipbuf_get_last_header(uip_buf, uip_len, &proto);
    is_me = uip_ds6_is_my_addr(&UIP_IP_BUF->srcipaddr);
    LOG_INFO("Outgoing packet (%s) proto %d to", is_me?
"send":"fwd", proto);
    LOG_INFO_6ADDR(&UIP_IP_BUF->destipaddr);
```

```
        LOG_INFO_("\n");
        return NETSTACK_IP_PROCESS;
    }
    struct netstack_ip_packet_processor packet_processor = {
        process_input = ip_input;
        process_output = ip_output;
    };
    PROCESS_THREAD(ipV6_hooks_process, ev, data)
    {   PROCESS_BEGIN();
        netstack_ip_packet_processor_add(&packet_processor);
        PROCESS_END();
}
```

Conclusion:

We understood the working of IPv6 in low power lossy network.

References:

→ https://ns3simulator.com
→ https://lowpowerlossynetwork.com

Term - Work 10

Problem Statement:
Understanding the working of IoT routing using RPL protocol.

Theory:

RPL is a distance vector-routing protocol. It mainly targets collection based networks, whose nodes periodically send measurements to a collection point. RPL provides a mechanism to disseminate information over the dynamically formed network topology

A DODAG is a DAG sorted at a single destination. The DODAG root has no outgoing edges. A DODAG is uniquely identified by a combination of RPI instance ID and DOBAGID

A nodes rank defines the nodes' individual position relative to other nodes with respect to DODAG root. Rank strictly increases in the down direction and strictly decreases in up-direction.

When going up, always forward to lower rank when possible, may forward to sibling if no lower rank exists

DIO DODAG information object

DIS_DODAG information solicitation

DAO- destination advertisement object.

Source Code:

```
#include "contiki.h"
#include "net/routing/routing.h"
#include "random.h"
#include "net/*"
#include "sys/log.h"
#define LOG_MODULE "App"
#define LOG_LEVEL LOG_LEVEL_INFO
#define WITH_SERVER_REPLY 1
#define UDP_CLIENT_PORT 8765
#define UDP_SERVER_PORT 5678
#define SEND_INTERVAL (60* CLOCK_SECOND)
static struct simple_udp_connection udp_conn;
PROCESS(udp_client_process, "UDP client");
AUTOSTART_PROCESSES(&udp_client_process);
static void udp_rx_callback(struct simple_udp_connection *c, const
uip_ipaddr_t *sender_addr, uint16_t sender_port, const uip_ipaddr_t
*receiver_addr, uint16_t receiver_port, const uint8_t *data, uint16_t datale
{  LOG_INFO("Received response '%.*s' from ", datalen, (char*) data);
   LOG_INFO_6ADDR(sender_addr);
   #if LLSEC802154_CONF_ENABLED
      LOG_INFO(" LLSEC LV:%d", uipbuf_get_attr(UIPBUF_ATTR_LLSEC_LEVE
   #endif  LOG_INFO_("\n");
}
PROCESS_THREAD(udp_client_process, ev, data)
{  static struct etimer periodic_timer;
   static unsigned count;
   static char str[32];
   uip_ipaddr_t dest_ipaddr;
   PROCESS_BEGIN();
```

```
    simple_udp_register(&udp_conn, UDP_CLIENT_PORT, NULL, UDP_SERVER_
PORT, udp_rx_callback);
    etimer_set(&periodic_timer, random_rand()%SEND_INTERVAL);
    while(1) {
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer));
        if(NETSTACK_ROUTING.node_is_reacheable() && NETSTACK_ROUTING
get_root_ipaddr(&dest_ipaddr)) {
            LOG_INFO("Sending request %u to ", count);
            LOG_INFO(&dest_ipaddr);
            LOG_INFO_("\n");
            snprintf(str, sizeof(str), "hello %d", count);
            simple_udp_sendto(&udp_conn, str, strlen(str), &dest_ipaddr)
            count++;
        } else {
            LOG_INFO("Not reacheable yet\n");
        }
        etimer_set(&periodic_timer, SEND_INTERVAL-CLOCK_SECOND+(random
rand()%(2*CLOCK_SECOND)));
    }
    PROCESS_END();
}
```

Server.c

```
#include "contiki.h"
#include "net/*"
#include "sys/log.h"
#define LOG_MODULE "App"
#define LOG_LEVEL LOG_LEVEL_INFO
#define WITH_SERVER_REPLY 1
#define UDP_CLIENT_PORT 8765
```

```
#define UDP_SERVER_PORT 5678
static struct simple_udp_connection udp_conn;
PROCESS(udp_server_process, "UDP server");
AUTOSTART_PROCESSES(&udp_server_process);
static void udp_rx_callback(struct simple_udp_connection *c, const
uip_ipaddr_t * sender_addr, uint16_t sender_port, const uip_ipaddr_t
*receiver_addr, uint16_t receiver_port, const uint8_t *data, uint16_t datalen)
{   LOG_INFO("Received request '%.5s' from ", datalen, (char*) data);
    LOG_INFO(sender_addr);
    LOG_INFO_("\n");
    #if WITH_SERVER_REPLY
        LOG_INFO("Sending response.\n");
        simple_udp_sendto(&udp_conn, data, datalen, sender_addr);
    #endif
}
PROCESS_THREAD(udp_server_process, ev, data)
{   PROCESS_BEGIN();
    NETSTACK_ROUTING.root_start();
    simple_udp_register(&udp_conn, UDP_SERVER_PORT, NULL, UDP_CLI
-PORT, udp_rx_callback);
    PROCESS_END();
}
```

Conclusion:

We understood the working of IoT routing using RPL protocol

References:

→ https://RPLprotocol.com